hTag: A Label-Based Extension for Hierarchical File Systems

James Kasten and Randy Yao

University of Michigan

Abstract

The traditional hierarchical file system is insufficient for meeting the behavioral needs of the typical user. As a result, non-hierarchical systems have been proposed. However, existing software, familiar habits, and various miscellaneous cases suggest that transitioning to a fully non-hierarchical model could prove disastrous. In this paper, we propose hTag, a simple and flexible label and tagging extension that sits on top of the existing hierarchical file system. hTag provides a system in which users can define labels with which files may be tagged while preserving the expected behavior of the native host file system. In doing so, we aim to provide an extension that seamlessly deploys into existing systems, providing users with the easy organizational benefits of a search and index driven model, while maintaining all of the benefits of their existing system. Furthermore, we demonstrate that this can be accomplished with a minimal footprint on disk and performance by careful bookkeeping and linking.

Background and Motivation

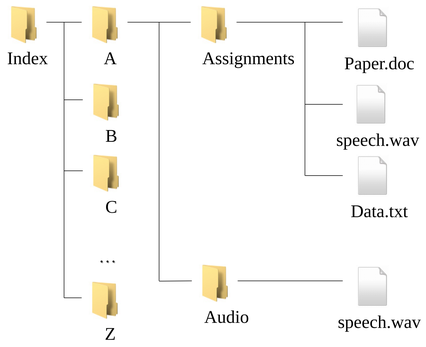
Users have been upgrading their hard drives with larger hard disks and storing more and more files in their systems. Previous work has argued that this trend has made hierarchical systems unfit for modern usage. It has been suggested that systems move towards a file systems that place more emphasis on indexing and searching. In an age where we search for everything in our web applications, this seems to be an appropriate claim.

Techniques to provide better search capabilities in FS

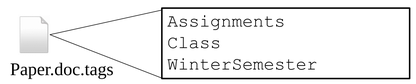
- Database on Hierarchical File System
  - Too slow – lots of memory references
  - Requires configuration which is too difficult for normal users
- Search tools built on hierarchical file system - (Ex. Spotlight)
  - Relies on known file types and info within the file
  - No additional input from user regarding context
- File system built on database
  - Not backwards compatible
  - Poor performance for particular workloads
- Proposed Index based tagged file system – Seltzer
  - Difficulty in maintaining backwards compatibility
  - Lose oftentimes beneficial hierarchical system
- hTag
  - Easy adoption
  - Data type independent
  - All the search benefits of a tagged file system
  - Allows both automated and user controlled labeling

We present hTag, a file system extension that aims to provide the benefits of a pure index based tag file system while also maintaining the benefits of hierarchical systems.

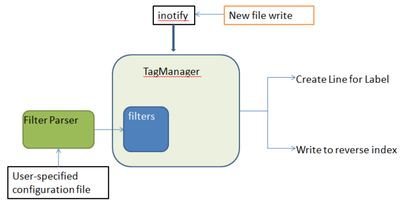
Implementation



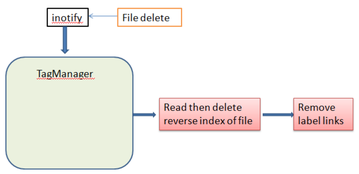
The hTag labeling system rests on top of a traditional hierarchical file system. The labels are represented as directories in the label index and all associated files have links within the directory.



In order to facilitate better performance when looking up a file's associated labels. hTag stores a reverse index in plain text document. This makes the cleanup caused by file deletions quick and efficient.



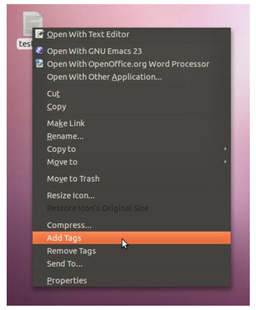
hTag utilizes Linux's inotify to watch for new files written to disk. The tagManager Python script keeps track of user-defined filters and adds the appropriate label to the file on a pattern match.



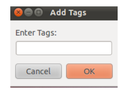
hTag watches for files deleted in the specified directories in order to clean up any links in the label index. This ensures no links to files are left behind after deletion.

The only other overhead the system incurs is during file renaming. Similar to the other events, hTag is notified when a file is renamed and all the file's tag links are updated appropriately.

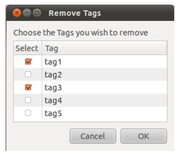
User Interface



Upon right-clicking a file the user is presented with the option to add and remove tags



The user can label the file with multiple tags at once



The remove tags option allows the user to quickly select and detach any undesired labels

In addition to the graphical user interface , hTag provides several command line scripts to make tagging, interacting and maintaining the system easy. As an example, hTag allows users to search list all files associated with a label with the lstag script, which can be written as a simple two-line wrapper to the ls utility.

Evaluation

Previous attempts at nontraditional models have expressed concerns in backwards compatibility and data agnosticism. We address these very easily with the fact that we're building entirely on top of a hierarchical system.

Cost of File Creations w/ single core

Files	10	50	100	200	500
No File Hooks	0.032	0.086	0.128	0.204	0.446
with TagManager	0.056	0.147	0.255	0.457	1.089

Cost of File Creations w/ two cores

Files	10	50	100	200	500
No File Hooks	0.025	0.063	0.132	0.328	0.913
with TagManager	0.033	0.068	0.124	0.244	0.554

We conducted a test showing the total performance cost when we create many files and have hTag's filter catch and tag every file with one tag. Although there is a performance penalty, we argue that it is unnoticeable in everyday use especially on multicore computers.

Disk usage

Tags	1000	5000	10000	20000	50000
Disk usage (MB)	0.3	1.4	2.83	5.65	14.15

The total disk usage is minimal. Even with a somewhat unrealistic volume of 50,000 file tags, our system only takes an additional 14.15MB on disk.